LAMPS
Internet-Draft
Intended status: Standards Track
Expires: 16 August 2022

M. Ounsworth
J. Gray
S. Mister
Entrust

12 February 2022

# Composite Encryption For Use In Internet PKI
## draft-ounsworth-pq-composite-encryption-01

Abstract

With the widespread adoption of post-quantum cryptography will come the need for an entity to possess multiple public keys on different cryptographic algorithms. Since the trustworthiness of individual post-quantum algorithms is at question, a multi-key cryptographic operation will need to be performed in such a way that breaking it requires breaking each of the component algorithms individually. This requires defining new structures for holding composite encryption data.

This document defines a content encryption process following the hybrid model as described in the NIST Post-Quantum Crypto FAQ. This draft defines three composite encryption modes. First, Composite Key Transport using Encryption primitives which encrypts a message (typically a content encryption key) for a recipient with a composite public key composed entirely of encryption keys by encrypting it with multiple one-time-pad keys, each encrypted under a different recipient public key. Second, Composite Key Transport using Encryption and KEM primitives is the generalization of the previous mode to support a mixture of encryption and KEM algorithms. Third, Composite Key Exchange is the most general and supports establishing a shared secret using any combination of encryption, KEM, and key exchange primitives where a master shared secret is generated using NIST SP 800-56Cr2.

This Internet-Draft will expire on 16 August 2022.



Table of Contents

## 1. Introduction

During the transition to post-quantum cryptography, there will be uncertainty as to the strength of cryptographic algorithms; we will no longer fully trust traditional cryptography such as RSA, Diffie-Hellman, DSA and their elliptic curve variants, but we will also not fully trust their post-quantum replacements until they have had sufficient scrutiny. Unlike previous cryptographic algorithm migrations, the choice of when to migrate and which algorithms to migrate to, is not so clear. Even after the migration period, it may be advantageous for an entity's cryptographic identity to be composed of key pairs associated with different public-key algorithms.

The deployment of composite public keys and composite encryption using post-quantum algorithms will face two challenges:

* Algorithm strength uncertainty: During the transition period, some post-quantum signature and encryption algorithms will not be fully trusted, while the trust in legacy public key algorithms will start to erode. A relying party may learn some time after deployment that a public key algorithm has become untrustworthy, but in the interim, they may not know which algorithm an adversary has compromised.

* Backwards compatibility: During the transition period, post-quantum algorithms will not be supported by all clients.

This document provides mechanisms to address algorithm strength uncertainty by building on ~~ reference draft-ounsworth-pq-composite-pubkeys ~~ by providing formats for both wrapping a content encryption key using multiple public key encryption mechanisms, or performing key exchange using a combination of encryption, key encapsulation, and key exchange primitives. The issue of backwards compatibility is addressed with support for Composite Or recipient keys in each mode.

This document is intended for general applicability anywhere that content encryption or key exchange is used.

### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

ALGORITHM: An information object class for identifying the type of cryptographic operation to be performed. This document is primarily concerned with algorithms for producing encryption keys.

BER: Basic Encoding Rules (BER) as defined in [X.690].

COMPONENT ALGORITHM: A single basic algorithm which is contained within a composite algorithm.

COMPOSITE ALGORITHM: An algorithm which is a sequence of one or more component algorithms..

DER: Distinguished Encoding Rules as defined in [X.690].

PUBLIC / PRIVATE KEY: The public and private portion of an asymmetric cryptographic key, making no assumptions about which algorithm.

PRIMITIVE PUBLIC KEY / SIGNATURE: A public key or signature object of a non-composite algorithm type.

SIGNATURE: A digital cryptographic signature, making no assumptions about which algorithm.

SECRET or SHARED SECRET: Cryptographic material established between two parties. May be generated by one party and send encrypted to the other, or may be the output of an exchange of public information between two or more parties that generates a unique shared value for all involved parties.

KEY DERIVATION FUNCTION: A function used to derive secure secret keys using shared secrets, hashing and other cryptographic primitives.

COMPOSITE ENCRYPTION KEY: A structure that contains a sequence of content encryption keys, or secrets used to derive a content encryption keys.

## 2. Composite Key Transport using Encryption primitives

In this composite encryption mode, a message to be encrypted is provided by the calling application. This message to be encrypted is assumed to have length less than the maximum message size of the chosen encryption algorithms, as is the case when a suitably-sized symmetric key is encrypted.

This mode is compatible with protocols requiring a key transport primitive, such as CMS' KeyTransRecipientInfo [RFC5652].

Composite Key Transport using Encryption primitives uses a trivial XOR one-time-pad scheme, as defined in Section 2.4. It transports n one-time-pad secret keys of the same length as the content to be encryption, where n is the number of recipient component public keys, and each one-time-pad secret key is encrypted under a different recipient component public key. The trivial XOR key-sharing scheme requires the recipient to use all component private keys in order to recover the content encryption key. Note that it would be possible to use an "n of m" or "threshold" secret sharing scheme if it was desired for the recipient to be able to complete the key transport using a subset of their private keys, but that mechanism is not defined in this document.

EDNOTE: we have not been able to find a reference and security analysis for the trivial XOR key-sharing scheme. This may need review by CFRG. We could re-frame this process as "a one-time pad with n-1 one-time pad keys, which we transport using the recipients public keys", then this could leverage one-time pad security analysis.

Composite encryption uses the following structure:

EDNOTE: Should a different composite OID be used to determine the type of composite encryption (Key Transport or Key Agreement?). Probably not because the desired key usage will be handled in the protocols that uses this privitive.

```
CompositeEncryptedKey ::= EncryptedKey{ SEQUENCE SIZE (2..Max)  OF OCTET STRING}
```

EDNOTE: This ASN.1 probably does not compile. The intent is that this fits into any EncryptedKey field, but defines some structure within the existing EncryptedKey ::= OCTET STRING, but I'm not sure exactly how to specify that.

Where each OCTET STRING within the SEQUENCE contains an encrypted one-time-pad secret key encrypted under one of the recipient component public keys. The CompositeEncryptedKey MUST list encrypted values in the same order as the recipient public key's component keys.

### 2.1. Algorithm Identifier

The id-alg-composite-encryption object identifier MUST be used to identify the usage of this mode

```
id-alg-composite-encryption OBJECT IDENTIFIER ::= {
    id-alg-composite-encryption OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
    Algorithm(80) Composite(4) id-alg-composite-encryption(4) }
```

EDNOTE: this is a temporary OID for the purposes of prototyping. Permanent OIDs should be requested from IANA, see Section 6.

### 2.2. Public key and key usage

The recipient MUST have a composite public key which supports key transport operations. Where the recipient public key has an associated keyUsage as specified in [RFC5280], it MUST have keyUsage: keyEncipherment. In other words, the mechanism specified in this section applies only if all of the recipient's public keys are associated with encryption algorithms.

#### 2.2.1. Composite-OR

The design intent of this mode is to support migration scenarios where a recipient has been provisioned with a composite key containing algorithms that its peers may not yet support. This mode allows the sender to encrypt for a subset of the recipient's public keys. Support for Composite OR subset encryption is indicated by the recipient at key generation time by marking its composite key with the id-composite-or-key algorithm identifier as defined in ~~~cite properly draft-ounsworth-pq-composite-keys~~~. To maximize security strength of the ciphertext, clients SHOULD encrypt for as many keys as they support and as the migration and compatibility situation allow.

Policy mechanisms defining allowed subets of algorithms could be applied here, but are out of scope of this document. As defined in this document, a recipient marking their public key as id-composite-or-key must accept the risk that a sender may encrypt sensitive data for it using any one of its component keys in isolation. Composite Or is a direct tradeoff of lower security for increased migration flexibility.

### 2.3. Algorithm parameters

The composite key transport using encryption mode does not require additional parameters, and therefore any associated Params are ABSENT.

### 2.4. Encryption process

The process for performing Composite Key Transport using Encryption primitives is as follows:

The first n-1 one-time-pad keys are random bit strings of the same length as the content encryption key. The final one-time-pad key is computed by XOR'ing the content encryption key with each of n-1 previous keys.

Input:

| | |
|---|---|
| n | The number of recipient component public keys |
| P1, P2, .., Pn | Recipient component public keys |
| A1, A2, .., An | Cryptographic algorithms to be used with public keys P1, P2, .., Pn |
| CEK | The Content Encryption Key |
| SIZE | The size of the Content Encryption Key in bits |

Output:

| | |
|---|---|
| E1, E2, .., En | EncryptedKey values corresponding to each recipient component public key |

Intermediate values:

| | |
|---|---|
| S1, S2, .., Sn-1 | One-time-pad secret keys to be encapsulated by each component algorithm |
| C | One-time-pad ciphertext of the CEK under S1, S2, .., Sn-1 |

Generation Procedure:

1. If recipient public key is of type id-composite-or-key, determine the

```
index of the last recipient public key to be encrypted for
  i_last := index of last Pi to be encrypted for
Else,
  i_last = n
```

2. To generate secret keys Sn, compute the following

```
C = CEK
for i := 1 to n

  a. If id-composite-or-key and Pi is to be skipped
    Ei := emptyOctetString
    continue to next i

  b. If i == i_last
      Ei = encrypt(C, Pi, Ai)
      break
    Else,
      Si := random_bits(SIZE)
      C := C XOR Si
      Ei = encrypt(Si, Pi, Ai)
```

3. Output E1, E2, .., En

Where random_bits(SIZE) is a cryptographically-secure random bit generator outputting SIZE bits, and where emptyOctetString is the octet string of length 0.

EDNOTE: we currently do not define a composite algorithmID type to carry A1, A2, .., An. We may need to add one analogously to the CompositeParams ::= SEQUENCE SIZE (2..MAX) OF AlgorithmIdentifier that we have in the composite signutares draft.

If the sender does not support Composite Or encryption, this algorithm may be simplified by omitting step 1, 2a, and the if i == i_last statement in 2b.

The design intent is that Composite Or encryption with a single recipient key collapses to being equivalent to direct encryption of the CEK.

### 2.5. Decryption process

To obtain the content-encryption key from a CompositeEncryptedKey, each component algorithm MUST be used to decrypt the set of one-time-pad keys. The keys are then XOR'ed together to recover the content encryption key.

```
Input:
     n                  The number of recipient component public keys

     SK1, SK2, .., SKn  Recipient component secret keys


     A1, A2, .., An     Cryptographic algorithms to be used with
                          public keys P1, P2, .., Pn

     E1, E2, .., En     EncryptedKey values corresponding to each recipient
                        component public key


Intermediate values:
     S1, S2, .., Sn     One-time-pad keys and ciphertext to be decapsulated
                        by each component algorithm

Output:
     CEK  The Content Encryption Key


Generation Procedure:
   1. Recover each one-time-pad key
        for i := 1 to n
          if Ei == emptyOctetString
            Si := emptyOctetString
          Else,
            Si := decrypt(Ei, SKi)

   2. Recover the CEK;
      For each one-time-pad key
        CEK = S1
        for i := 2 to n
           if Si != emptyOctetString
             CEK = CEK XOR Si

   3. Output CEK
```

The if statement in step 1 (and ensuring its proper bit length for the XOR in step 2) is the only modification required to support Composite Or encryption. The designers have intentionally omitted a check that the recipient key is of type id-composite-or-key because even if the sender erroneously used composite or subset encryption for a recipient key which is not of type id-composite-or-key, the damage has already been done by encrypting and transmitting the data, no further harm can be done by decrypting it. However, where appropriate, clients SHOULD indicate a warning to users that this data was transmitted with weaker encrypting than their public key allows.

EDNOTE: investigate whether this is actually a special case of the next mechanism, and therefore both sections can be folded together.

## 3. Composite Key Transport using Encryption and KEM primitives

This composite encryption mode is the generalization of the mode defined in Section 2 to support a composite recipient public key which may contain a mixture of one or more encryption component algorithms with zero or more key encapsulation mechanism (KEM) component algorithms.

This mode is compatible with protocols requiring a key transport primitive, such as CMS' KeyTransRecipientInfo [RFC5652].

Security consideration: for a recipient composite public key to be applicable to this mode, all component KEMs MUST produce a shared secret whose bits are independent and uniformly distributed (aka "uniformly IID" or "uniformly random" or "full entropy") and therefore the shared secret is safe to use direcly as a symmetric key. If a recipient public key contains component KEMs which are not know to have this property, then implementors SHOULD use the more general mode described in Section 4 which incorporates the use of a key derivation function. See Section 7.1 for a further discussion of this security consideration.

EDNOTE: also put this in the Security considerations section.

### 3.1. Algorithm Identifier

The id-alg-composite-kem object identifier MUST be used to identify the usage of this mode

```
id-alg-composite-kem OBJECT IDENTIFIER ::= {
    id-alg-composite-encryption OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
    Algorithm(80) Composite(4) id-alg-composite-kem(5)}
```

EDNOTE: this is a temporary OID for the purposes of prototyping. Permanent OIDs should be requested from IANA, see Section 6.

### 3.2. Public key and key usage

The recipient MUST have a composite public key which supports key transport or key encapsulation operations. Where the recipient public key has an associated keyUsage as specified in [RFC5280], it MUST have keyUsage: keyEncipherment. In other words, the mechanism specified in this section applies only if all of the recipient's public keys are encryption or KEM algorithms.

In addition, for a recipient composite public key to be applicable to this mode, all component KEMs MUST be capable of producing a shared secret of SIZE bits, where SIZE is the length in bits of the content encryption key (CEK) to be transported. This is assumed for the remainder of this section.

#### 3.2.1. Composite-OR

The design intent of this mode is to support migration scenarios where a recipient has been provisioned with a composite key containing algorithms that its peers may not yet support. This mode allows the sender to encrypt for a subset of the recipient's public keys. Support for Composite OR subset encryption is indicated by the recipient at key generation time by marking its composite key with the id-composite-or-key algorithm identifier as defined in ~~~cite properly draft-ounsworth-pq-composite-keys~~~.

Policy mechanisms defining allowed subets of algorithms could be applied here, but are out of scope of this document. As defined in this document, a recipient marking their public key as id-composite-or-key must accept the risk that a sender may encrypt sensitive data for it using any one of its component keys in isolation. Composite Or is a direct tradeoff of lower security for increased migration flexibility.

### 3.3. Algorithm parameters

The composite key transport using encryption and KEM mode does not require additional parameters, and therefore any associated Params are ABSENT.

### 3.4. Encryption process

Given these conditions are met, the encryption process defined in Section 2.4 is modified as follows:

```
Input:
     n                  The number of recipient component public keys
     P1, P2, .., Pn     Recipient component public keys
     CEK                The Content Encryption Key
     SIZE               The size of the Content Encryption Key in bits

Output:
     E1, E2, .., En     EncryptedKey values corresponding to each recipient
                        component public key

Intermediate values:
     S1, S2, .., Sn     One-time-pad secret keys to be encapsulated by each
                        component algorithm

Generation Procedure:

   1. If recipient public key is of type id-composite-or-key, determine the
      index of the last recipient public key to be encrypted for
        i_last := index of last Pi to be encrypted for
      Else,
        i_last = n


   2.
      for i := 1 to n
        a. if id-composite-or-key and Pi is to be skipped
            Ei := emptyOctetString
            continue to next i

        b. If i == i_last
            continue to next i
          Else, if Pi is of type KEM:
            Si,Ei := encaps(Pi)
            CEK := CEK XOR Si
          Else:
            Si := random_bits(SIZE)
            CEK := CEK XOR Si
            Ei := encrypt(Si, Pi)


   3. Encrypt the final CEK value
            Ei_last = encrypt(CEK, Pi_last)

   4. Output E1, E2, .., En

   Where random_bits(SIZE) is a cryptographically-secure random bit
   generator outputting SIZE bits, and where emptyOctetString is the
   octet string of length 0.
```

If the sender does not support Composite Or encryption, this algorithm may be simplified by omitting step 1, 2a, and the if i == i_last statement in 2b.

The design intent is that Composite Or encryption with a single recipient key collapses to being equivalent to direct encryption of the CEK.

### 3.5. Decryption process

The decryption process defined in Section 2.5 applies directly where decrypt() is substituted for decaps() when the underlying primitive is a KEM.

## 4. Composite Key Exchange

This mode is the most general in that it supports a composite recipient public key which MAY contain an arbitrary mixture of encryption, key encapsulation mechanism (KEM), and key agreement component algorithms. Due to the nature of key agreement algorithms, this mode cannot take a content encryption key as input, but instead generates a master shared secret as an output. As such, the nomenclature in this mode differs from the modes above.

This mode is compatible with protocols requiring a key agreement primitive, such as CMS' KeyAgreeRecipientInfo [RFC5652].

Composite key exchange uses the underlying primitive to either encrypt for, encapsulate, or interactively do key agreement with each of the recipient's public keys, then all shared secrets are concatenated together and a KDF is applies as prescribed by NIST SP 800-56Cr2 [SP80056cr2].

### 4.1. Algorithm Identifier

The id-alg-composite-keyex object identifier MUST be used to identify the usage of this mode

```
id-alg-composite-keyex OBJECT IDENTIFIER ::= {
    id-alg-composite-encryption OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
    Algorithm(80) Composite(4) id-alg-composite-encryption(6) }
```

EDNOTE: this is a temporary OID for the purposes of prototyping. Permanent OIDs should be requested from IANA, see Section 6.

### 4.2. Public key and key usage

The recipient MUST have a composite public key which supports key transport, key encapsulation, or key exchange operations. Where the recipient public key has an associated keyUsage as specified in [RFC5280], it MUST have keyUsage: keyEncipherment, keyAgreement. This mode is the most general and places the fewest restrictions on the recipient public key.

EDNOTE: I think this violates our public key draft where we say that the public key's KU MUST apply to all components. ... we did not want mixing of signatures and encryption keys, but I think in this case we do want to allow mixing of keyEncipherment and keyExchange keys. Not sure how to fix that.

#### 4.2.1. Composite-OR

The design intent of this mode is to support migration scenarios where a recipient has been provisioned with a composite key containing algorithms that its peers may not yet support. This mode allows the sender to encrypt for a subset of the recipient's public keys. Support for Composite OR subset encryption is indicated by the recipient at key generation time by marking its composite key with the id-composite-or-key algorithm identifier as defined in ~~~cite properly draft-ounsworth-pq-composite-keys~~~.

Policy mechanisms defining allowed subets of algorithms could be applied here, but are out of scope of this document. As defined in this document, a recipient marking their public key as id-composite-or-key must accept the risk that a sender may encrypt sensitive data for it using any one of its component keys in isolation. Composite Or is a direct tradeoff of lower security for increased migration flexibility.

### 4.3. Algorithm parameters

The composite key exchange mode requires additional parameters to specify the KDF used to combine shared secrets into a master shared secret.

Params ::= KeyDerivationAlgorithmIdentifier

The KeyDerivationAlgorithmIdentifier type is specified in [RFC5652]. The KeyDerivationAlgorithmIdentifier definition is repeated here for completeness.

KeyDerivationAlgorithmIdentifier ::= AlgorithmIdentifier

### 4.4. Encapsulation Process

Composite key exchange uses the underlying primitive to either encrypt for, encapsulate, or interactively do key agreement with each of the recipient’s public keys, then all shared secrets are concatenated together and a KDF is applies as prescribed by NIST SP 800-56Cr2 [SP80056cr2].

```
Input:
     P1, P2, .., Pn     Public keys for the n component encryption
                        algorithms, a CompositePublicKey

    SIZE                The size, in bits, for shared secrets to be combined by b
oth
                        parties into a content encryption key. This value SHOULD
correspond
                        to the size of the content encryption key.

    KDF                 A key derivation function

Output:
     E1, E2, .., En     EncryptedKey values corresponding to each recipient
                        component public key

     M                  Master shared secret


Ciphertext and master secret Generation Procedure:
     1. Generate a set of one-time-pad secret keys of
      the same length as the content encryption key
        for i := 1 to n

      a. if id-composite-or-key and Pi is to be skipped
            Si = emptyOctetString
            Ei := emptyOctetString
            continue to next i

      b. if P1 is of type KEM or keyExchange:
            Si,Ei := encaps(Pi)
        else:
            Si := random_bits(SIZE)
            Ei := encrypt(Si, Pi)

   2. Generate Z via concatenation
        Z = S1 || S2 || .. || Sn

   3. Generate the master shared secret via a KDF
        M = KDF(Z)

   4. Output M
        Output E1, E2, .., En
```

Where emptyOctetString is the octet string of length 0 that serves as a no-op or identity element for the concatenation in step 2.

In cases where KDF is extensible output function, the length of M must be carried in the KeyDerivationAlgorithmIdentifier defined in Section 4.3.

EDNOTE: It isn't clear to us how one uses the defined HKDF algorithmid (RFC 8619) here. Those OIDs specify a hash, but no output length or seed or info parameter either implicitely or explicitely. But we also don't see how it would be used with CMS either, for the same reason. ..?

If the sender does not support Composite Or encryption, this algorithm may be simplified by omitting step 2a.

EDNOTE: investigate whether step 3 really belongs here, or whether the surrounding protocol (ex. CMS EnvelopedData) will perform a final KDF anyways. We believe that outputting an IID master secret is consistent with modern KEM behaviour.

### 4.5. Decapsulation Process

```
Input:
     n                  The number of recipient component public keys

     SK1, SK2, .., SKn  Recipient component secret keys

     E1, E2, .., En     EncryptedKey values corresponding to each recipient
                          component public key

     KDF                A key derivation function


Intermediate values:
     S1, S2, .., Sn     Shared secrets to be encapsulated by
                          each component algorithm

Output:
     M                  Master shared secret


 Master Secret Recovery Procedure:
   1. Recover each shared secret
        for i := 1 to n
          if Ei == null
            Si = EMPTY_STRING
          Si := decrypt_or_decaps(Ei, SKi)

   2. Generate Z via concatenation
        Z = S1 || S2 || .. || Sn

   3. Generate the master shared secret via a KDF
        M = KDF(Z)

  4. Output M
```

"EMPTY_STRING" indicates a string or byte array of length zero so that that value as essentially omitted from the concatenation in step 2.

The if statement in step 1 is the only modification required to support Composite Or encryption. The designers have intentionally omitted a check that the recipient key is of type id-composite-or-key because even if the sender erroneously used composite or subset for a recipient key which is not of type id-composite-or-key, the damage has already been done by generating a master secret and potentially transmitting data encrypted with it, no further harm can be done by decrypting it. However, where appropriate, clients SHOULD indicate a warning to users that this data was transmitted with weaker encrypting than their public key allows.

## 5. In Practice

This section addresses practical issues of how this draft affects other protocols and standards.

## 6. IANA Considerations

The following OIDs are to be assigned by IANA. The authors suggest that IANA assign OIDs for composite encryption on the id-pkix arc:

```
id-alg-composite OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) algorithms(6) composite(??) id-alg-composite-encryption
(??)}

id-alg-composite-kem OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) algorithms(6) composite(??) id-alg-composite-kem(??)}

id-alg-composite-keyex OBJECT IDENTIFIER ::= {
    iOBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) algorithms(6) composite(??) id-alg-composite-encryption
(??)}
```

## 7. Security Considerations

### 7.1. IID property of KEM primitives

Composite Key Transport using Encryption and KEM primitives defined in Section 3 directly uses the shared secret output from the underlying KEM primitevas as a one-time-pad key to encrypt the CEK. Therefore the output of the KEM primitive of needs to meet the security properties of a one-time-pad key, namely that its bits are independent and identically distributed (IID). In particular, key agreement schemes such as ECDH or SIKE do not produce shared secrets that meet this requirement and therefore MUST use the fully general mechanism Composite Key Exchange defined in Section 4.

EDNOTE: Should this be brough to CFRG to decide which KEMs are appropriate to use with this mechanism? It may be possible that we need to run the KEM output through a KDF; but we’re trying to avoid needing to carry a KDF AlgID here.

### 7.2. Composite-OR modes

Composite-OR eases migration at the expense of security. For composite encryption and key encapsulation, the weakening of security is entirely at the discretion of the sender, since once data has been encrypted and transmitted with weak ciphers, there is nothing the recipient can do to protect the data against record & decrypt attacks. Clients performing Composite Or encryption operations MUST ensure that the recipient's public key is of type id-composite-or-key before producing a ciphertext with a subset of the recipient's public keys.

For some cases of composite key exchange, notably when the underlying key exchange primitive is used in a fully interactive (aka "ephemeral-ephemeral") mode, the sender cannot begin encrypting data until the recipient has completed the key exchange. The recipient SHOULD reject the connection if one or more null ciphertexts are encountered when the recipient's public key is not of type id-composite-or-key.

### 7.3. Policy for Deprecated or Unacceptable Algorithms

Within the context of composite encryption, the sender holds the responsibility to ensure that chosen algorithms are of sufficient strength prior to encrypting and transmitting sensitive data under them. Composite is designed to provide security redundancy and to remain strong as long as at least one of the component algorithms remains strong.

When encrypting for a Composite-OR public key and using a subset of the recipient's public key, then these redundancy guarantees no longer apply. The sender SHOULD employ a policy mechanism to ensure that they are using a combination of algorithms of sufficient strength. Even though this document does not define such a policy mechanism, but implementors making use of Composite-OR encryption are strongly encouraged to implement a policy mechanism.

## 8. Appendices

### 8.1. ASN.1 Module

~~ TODO ~~

### 8.2. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

https://datatracker.ietf.org/ipr/3588/

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

### 8.3. Making contributions

Additional contributions to this draft are weclome. Please see the working copy of this draft at, as well as open issues at:

https://github.com/EntrustCorporation/draft-ounsworth-pq-composite-encryption

## 9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC3211] Gutmann, P., "Password-based Encryption for CMS", RFC 3211, DOI 10.17487/RFC3211, December 2001, <https://www.rfc-editor.org/info/rfc3211>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[SP80056cr2] NIST, "SP 800-56c Rev. 2: Recommendation for Key-Derivation Methods in Key-Establishment Schemes", August 2020.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2015, November 2015.

## Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com

John Gray
Entrust Limited

Email: john.gray@entrust.com

Serge Mister
Entrust Limited

Email: serge.mister@entrust.com